

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-068993

(43)Date of publication of application : 16.03.2001

(51)Int.Cl.

H03K 19/173

H01L 21/82

(21)Application number : 11-238384

(71)Applicant : FUJI XEROX CO LTD

(22)Date of filing : 25.08.1999

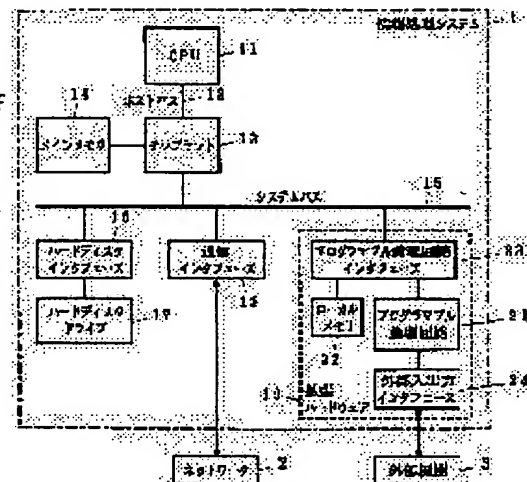
(72)Inventor : SATO YOSHIHIDE

(54) INFORMATION PROCESSING SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To reduce power consumption by relieving a load of a main processor to reconfigure a programmable logic circuit and to enhance the performance of the system as a whole.

SOLUTION: For example, a local memory 22 stores information of a circuit in use and a sequence in advance. Furthermore, a header part of processing data given from a CPU 11 includes information to specify first circuit information used to reconfigure a programmable logic circuit 21. A programmable logic circuit interface 23 interprets the header part to specify the first circuit information, a programmable logic circuit 21 reconfigures the processing circuit to process the processing data. Moreover, the processing circuit is reconfigured from the circuit information in the sequence stored in the local memory 22 to execute the processing in the processing circuit sequentially. Thus, the CPU 11 needs not conduct processing of the reconfiguration.



LEGAL STATUS

[Date of request for examination]

27.10.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号
特開2001-68993
(P2001-68993A)

(43) 公開日 平成13年3月16日 (2001.3.16)

(51) Int.Cl. ⁷	識別記号	F I	テーマコード(参考)
H 0 3 K 19/173	1 0 1	H 0 3 K 19/173	1 0 1 5 F 0 6 4
H 0 1 L 21/82		H 0 1 L 21/82	A 5 J 0 4 2
			C

審査請求 未請求 請求項の数7 O L (全 15 頁)

(21) 出願番号 特願平11-238384

(22) 出願日 平成11年8月25日 (1999.8.25)

(71) 出願人 000005496

富士ゼロックス株式会社

東京都港区赤坂二丁目17番22号

(72) 発明者 佐藤 嘉秀

神奈川県足柄上郡中井町境430 グリーン

テクノカい 富士ゼロックス株式会社内

(74) 代理人 100101948

弁理士 柳澤 正夫

Fターム(参考) 5F064 AA07 BB09 BB12 BB40 DD07

FF04 FF36 HH05

5J042 AA10 BA01 BA02 BA11 CA00

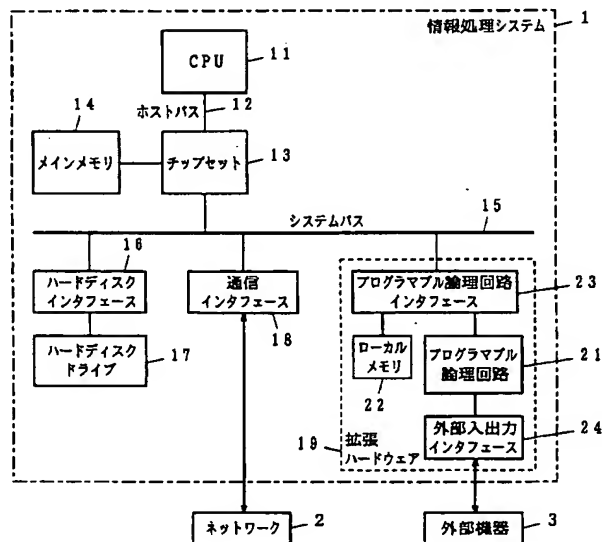
CA20 DA02 DA03 DA04

(54) 【発明の名称】 情報処理システム

(57) 【要約】

【課題】 プログラマブル論理回路を再構成するためのメインプロセッサの負荷を大幅に軽減させ、低消費電力化を図るとともに、システム全体のパフォーマンスを向上させた情報処理システムを提供する。

【解決手段】 例えばローカルメモリ22には、予め使用する回路情報と順番が保持されている。また、CPU11から与えられる処理データのヘッダ部に、プログラマブル論理回路21の再構成に使用する最初の回路情報を特定する情報を含んでいる。プログラマブル論理回路インタフェース23は、ヘッダ部を解釈して最初の回路情報を特定し、プログラマブル論理回路21に処理回路を再構成し、処理データの処理を行わせる。さらに、ローカルメモリ22に保持されている順番で、回路情報から処理回路を再構成し、その処理回路での処理の実行を、順次行わせる。これによって、CPU11は再構成の処理を行わなくて済む。



【特許請求の範囲】

【請求項 1】 回路情報を変更することによって機能を随時変更し再構成することが可能なプログラマブル論理回路を備えた情報処理システムにおいて、予め使用する回路情報と順番を保持する回路情報保持手段と、与えられた処理データから前記プログラマブル論理回路の再構成に使用する最初の回路情報を特定するとともに特定された回路情報から前記回路情報保持手段に保持されている順番に前記回路情報により前記プログラマブル論理回路を再構成して処理回路を構成し該処理回路に前記処理データの処理を行わせるインタフェース手段を有していることを特徴とする情報処理システム。

【請求項 2】 前記処理データのヘッダ部には、あらかじめ規定された処理ステップの順番、前記順番に対応した再構成する回路情報を格納した前記回路情報保持手段のアドレス情報、前記順番に対応した回路で処理されたデータを格納するための記憶手段のアドレス情報、前記順番に対応して前記プログラマブル論理回路に連続的に同時再構成可能な回路数の情報が付加されていることを特徴とする請求項 1 に記載の情報処理システム。

【請求項 3】 前記インタフェース手段は、前記プログラマブル論理回路の構成可能な最大回路規模と再構成する回路規模との関係において、連続する処理回路が同時に再構成できる領域が確保できない場合に、処理ステップの順番に対応して回路の再構成と構成した処理回路による処理と、該処理回路で処理された中間処理データの前記記憶手段への入出力を順次行いながら連続的に処理が行われるように制御することを特徴とする請求項 2 に記載の情報処理システム。

【請求項 4】 前記インタフェース手段は、前記プログラマブル論理回路の構成可能な最大回路規模と再構成する回路規模との関係において、複数の処理回路の再構成を行ってから、該処理回路における処理が連続して実行され、最後の処理回路による結果を中間処理データとして前記記憶手段に格納し、続けて複数の処理回路の再構成と複数の処理回路における処理が順次実行されるように制御することを特徴とする請求項 2 に記載の情報処理システム。

【請求項 5】 前記インタフェース手段は、前記プログラマブル論理回路の構成可能な最大回路規模と再構成する回路規模との関係において、複数の処理回路の再構成を、同時に再構成できる数だけ連続的に行いながら、最初の処理回路の再構成が終了した時点で最初の処理が実行されるとともに、処理の実行と並列的に次の回路の再構成を行い、処理結果を次に再構成された処理回路に受け渡して連続して処理が行われるように制御することを特徴とする請求項 2 に記載の情報処理システム。

【請求項 6】 前記処理回路における処理結果は中間処理データとして前記記憶手段に格納可能であることを特徴とする請求項 4 または請求項 5 に記載の情報処理シ

テム。

【請求項 7】 前記処理回路における処理結果は、予め規定した処理ステップの処理終了ごとに中間処理データとして前記記憶手段に格納可能であることを特徴とする請求項 4 または請求項 5 に記載の情報処理システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、アプリケーションプログラムによる処理の一部分を、回路構成を再構成できるプログラマブル論理回路で処理することが可能である情報処理システムに関する。特に、回路の再構成と処理を連続的に実行する方法に関するものである。

【0002】

【従来の技術】デジタル回路装置、特に特定用途向け集積回路（ASIC）の分野において、製品の開発期間を短縮するために、フィールドプログラマブルゲートアレイ（FPGA）やプログラマブルロジックデバイス（PLD）などで構成されたプログラマブル論理回路が広く使われている。これらのプログラマブル論理回路は、論理回路を記述する回路情報を読み込ませることで、内部の論理回路と論理回路間の結線を自由に構成することができる。このため、プログラマブル論理回路を用いることで、従来は回路設計の終了後に数週間から数か月を必要とした集積回路の作製時間が不要となるメリットがある。特に、例えば米国特許第 4,700,187 号明細書等に記載されているような電氣的に再構成可能なプログラマブル論理装置は、一度作製した回路を必要に応じて自由に何度でも変更できるという利点があり、ますます広く使われるようになってきている。

【0003】ところで、最近の論理回路はますます複雑になってきており、ひとつのプログラマブル論理回路では実現できない規模にまで回路規模が大きくなっている。この問題を解決するためのひとつの方法として、異なる時間に異なる論理回路を実現するために、プログラマブル論理回路を処理の途中で再構成することが提案されている。この方法を用いることにより、携帯情報端末のように装置が小型であるために内蔵できる回路規模に制約がある場合でも、様々な処理を比較的高速に行うことができるという利点がある。

【0004】しかし、プログラマブル論理回路を再構成するときには、回路全体の回路情報を再度読み込ませる必要があるため、再構成に時間がかかるという欠点がある。さらに、処理の途中で再構成することは、処理を一時中断し、その時のデータをプログラマブル論理回路の外部の記憶装置に待避させ、新たな回路情報を読み込んで再構成し、再構成前のデータと再構成に伴う新しいデータを入力するという余分な処理が必要になる。

【0005】この問題を解決するものとして、例えば米国アトメル社の「CONFIGURABLE LOGIC」という名のデータブックに記載されているプログラ

マブル論理回路、および米国ザイリンクス社の「THE PROGRAMMABLE LOGIC」という名のデータブックに記載されているプログラマブル論理回路等がある。これらのプログラマブル論理回路は、データを記憶するためのデータ記憶装置を有し、回路の動作中でも外部の記憶装置から回路情報の一部を読み込んで部分的に再構成を行うことができる。これによって、再構成するための時間を最小に留めるようにしている。

【0006】このようなプログラマブル論理回路を情報処理システムに用いる場合には、例えばCPUなどの制御装置がアプリケーション等の処理とプログラマブル論理回路の再構成とを高速かつ効率的に行わなければならないという問題がある。すなわち制御装置は、所望の論理回路を構成するための回路情報を格納先から取り出し、必要に応じて複数の回路情報を合成し、プログラマブル論理回路内に所望の論理回路を再構成する。これと並行して、アプリケーションの処理を実行する必要がある。両者の処理を高速かつ効率的に行わねばならない。

【0007】以上に述べた複数の回路情報によりプログラマブル論理回路を再構成しながら処理を行う情報システムは、ネットワークに接続して利用することができる。その例として、特開平10-78932号公報に公開される「リコンフィグラブル・ネットワークコンピュータ」がある。

【0008】図17は、従来の情報処理システムの一例を示すブロック図である。図中、51はアプリケーションサーバ、52、53はクライアントコンピュータ、54は通信ネットワーク、55はメインプロセッサ、56は拡張ハードウェア、57はアプリケーションプログラム、58は拡張コード、59はメインプロセッサコード、60はOS、61はコード選択機能である。この情報処理システムは、通信ネットワーク54によって接続された複数のコンピュータで構成されている。

【0009】アプリケーションサーバ51は、アプリケーションプログラムを配布するコンピュータである。また、クライアントコンピュータ52、53は、アプリケーションサーバ51からアプリケーションプログラム57をダウンロードして実行する。クライアントコンピュータ52、53には、メインプロセッサ55が搭載されており、OS60の管理下においてアプリケーションプログラム57を実行することができる。また、クライアントコンピュータ52は、メインプロセッサ55とは別に拡張ハードウェア56を有している。拡張ハードウェア56は、上述のプログラマブル論理回路が搭載されており、プログラムにより機能を随時変更し、再構成することが可能である。

【0010】アプリケーションサーバ51に格納されたアプリケーションプログラム57においては、その一部の機能については、拡張ハードウェア56のプログラムコード（拡張コード58）と、クライアントコンピュー

タのメインプロセッサ55のコード（メインプロセッサコード59）が含まれている。

【0011】また、クライアントコンピュータ52、53のOS60には、それぞれのハードウェア構成に適したコードを選択するコード選択機能61を有している。このコード選択機能61が、拡張ハードウェア56を実装しているか否かを判断し、クライアントコンピュータ52のように拡張ハードウェア56が実装されている場合には、アプリケーションプログラム57の中から拡張コード58を取り出して拡張ハードウェア56により実行する。またクライアントコンピュータ53のように拡張ハードウェアを持たない場合には、メインプロセッサコード59を選択して実行する。

【0012】別の構成では、拡張ハードウェア56で実現する機能を、クライアントコンピュータ上に後から動的に追加／削除が可能なOS60の拡張機能あるいは動的ライブラリとして実現し、アプリケーションプログラム57がOS60に対し、処理中に利用する拡張機能あるいは動的ライブラリの種類を登録する。OS60は、拡張機能あるいは動的ライブラリがクライアントコンピュータ上に存在する場合にはそれを用い、存在しない場合には通信ネットワーク54上のアプリケーションサーバ51から必要とする拡張機能あるいは動的ライブラリを転送して利用する。

【0013】また、メインプロセッサコード59と拡張コード58は、一体となっているのではなく、アプリケーションプログラム57またはOS60の拡張機能または動的ライブラリ毎に、個々のコードをアプリケーションサーバ51の上に備えている。

【0014】拡張ハードウェア56を構成するプログラマブル論理回路の構成が、クライアントコンピュータ間で異なる場合は、拡張コード58を、適当なゲート数と入出力端子数の論理回路の機能をブール式等で記述した基本モジュールと、それらの接続関係を表現したコードから構成される。この基本モジュールをそれぞれプログラマブル論理回路の基本プログラムとして割り付ける機能と、複数のプログラマブルロジックチップにまたがる大きな拡張コードの場合には、基本モジュールを接続の度合いに応じて分割し、各プログラマブルロジックチップに配置、配線する機能を、アプリケーションサーバ51またはクライアントコンピュータ上に持つ。

【0015】拡張ハードウェア56を利用する複数のアプリケーションを同時に実行できるように、必要のなくなったハードウェア資源を別のアプリケーションプログラムのために再利用するハードウェア資源の管理機能と、拡張ハードウェア56に入りきらない拡張コードを時分割で入れ替えるコード入れ替え機能を持つ。クライアントコンピュータ上で実行されるアプリケーションプログラム毎に適宜設定されるプライオリティ値、メインプロセッサ55の処理能力値、拡張ハードウェア56の

処理能力値、ハードウェア資源量、コードを入れ替えるために必要な処理能力値を基に、ハードウェア資源に入りきらない複数のアプリケーションプログラムに対して選択する拡張ハードウェア管理機能を持つ。複数のアプリケーションが同時に同じ拡張コードを拡張ハードウェア 56 で利用する場合には、内部状態のみを時分割で切り替えて機能を共有する。

【0016】以上のように、ネットワークで接続されたコンピュータ上で、アプリケーションサーバから配布されたアプリケーションプログラムをクライアントコンピュータ側で実行する際、プログラムにより機能を随時変更し、再構成可能な拡張ハードウェアをクライアントコンピュータに搭載する。そして、アプリケーションサーバに格納されたアプリケーションプログラムには、メインプロセッサコードと拡張コードを含ませておく。実行時には、拡張ハードウェアの有無、種類を判断する OS のコード選択機能によって、クライアントコンピュータ側で拡張ハードウェアの構成を変え、処理に適した構成にする。これによって、クライアントコンピュータにおいて、アプリケーションプログラムを高速に処理することができる。

【0017】また、従来、ネットワーク上で、クライアントコンピュータ側に特殊なハードウェアを必要とする新しいサービスを開始しようとする場合、クライアントコンピュータ側のユーザは、そのために新しいハードウェアを導入する必要があった。またサービスの提供者は、新しいハードウェアをもつ一部のユーザに対してのみ、新しいサービスを提供していた。しかし上述のように、プログラマブル論理回路を搭載した拡張ハードウェアを有している構成では、プログラマブル論理回路を再構成することによって、新しいハードウェアを導入することなく、新しいサービスを開始することが可能となる。

【0018】このように拡張ハードウェアにプログラマブル論理回路を用いて、回路を再構成しながら処理を実行していく場合には、CPU による制御が必要であった。例えばアプリケーションプログラムにおいて、処理のフローが決定されていて、そのために用いる機能回路があらかじめ選択指定可能な場合にも、常にメインプロセッサを動作させた処理方法が用いられている。特に、上述のシステム例のように、クライアントコンピュータに拡張ハードウェアが搭載されている場合、拡張コードによって再構成する制御は、クライアントコンピュータのメインプロセッサにより行われる。そのため、アプリケーションプログラムのメインプログラムにはあらかじめ必要な拡張コードが関連づけされていて、拡張ハードウェアの再構成の都度、メインプロセッサの制御によって拡張コードを用いて行われている。このため、多種多様な拡張ハードウェアを頻繁に再構成しながらアプリケーションプログラムを実行していく場合には、常に、メ

インプロセッサが処理の進行状態を監視するとともに、再構成のための制御を行う必要があり、メインプロセッサの負荷が大きくなる欠点がある。また、拡張ハードウェアの動作とともにメインプロセッサも動作させる必要があり、クライアント全体の消費電力も増大する欠点がある。

【0019】このように、回路の再構成のための制御によって、メインプロセッサの処理負荷が常時発生することにより、消費電力として、メインプロセッサと拡張ハードウェアとを合わせた増加となってしまう問題がある。また、回路の再構成のためのメインプロセッサの負荷のため、システム全体のパフォーマンスが制約を受けて低下してしまう欠点があった。

【0020】次に、プログラマブル論理回路の新しいデバイス技術について述べる。アプリケーションの処理に合わせた処理回路をプログラマブル論理回路上に構成し、この専用の処理回路を用いて高速処理を実現するというリコンフィギュラブルコンピューティングにプログラマブル論理回路が活用されはじめている。リコンフィギュラブルコンピューティングでは、アプリケーション処理で必要となる複数の処理回路の回路情報を記憶装置へ事前に格納しておき、必要に応じて記憶装置から読み出した回路情報をプログラマブル論理回路に書き込むことで、その時点で必要となる回路を生成する。この技術はキャッシュロジック技術とかバーチャルロジック技術と呼ばれる。

【0021】キャッシュロジック技術は、同じプログラマブル論理回路上に必要に応じて異なる回路を構成するという時分割駆動技術である。その結果、回路規模の小さなプログラマブル論理回路を用いて、その回路規模以上の回路を実現でき、回路装置の小型化と低コスト化が可能となる。しかしながら、プログラマブル論理回路に書きこむ回路情報の規模によっては、回路の再構成時間が長くなり、専用の処理回路を用いて高速処理を実現するというリコンフィギュラブルコンピューティングの効果損なうという問題がある。

【0022】この問題のひとつの解決方法が、マルチコンテキスト技術と呼ばれるデバイス技術である。すなわち、プログラマブル論理回路内に複数の回路情報を格納するメモリを備え、必要に応じてメモリを切り替えて回路を再構成することにより、回路の再構成時間を大幅に短縮できる。

【0023】マルチコンテキスト技術の従来例のひとつが、FPD' 95 の "A First Generation DPGA Implementation" で示された DPGA である。図 18 は、DPGA の論理セル構造の一例の説明図である。図中、71 は DRAM、72、75 はマルチプレクサ、73 はルックアップテーブル、74 はフリップフロップである。図 18 に示すように、DPGA の論理セルは、4 組の構成を格納する 4

×32ビットのDRAM71、4つの8入力マルチプレクサ72、4入力ルックアップテーブル73、フリップフロップ74、出力を切り替えるマルチプレクサ75で構成されている。32ビットのDRAM71の出力のうち、12ビットが4つの8入力マルチプレクサ72の状態を、16ビットが4入力ルックアップテーブル73の状態を、1ビットがマルチプレクサ75の状態を決定し、残り3ビットは保留されている。4組のDRAM71には、それぞれ異なるデータが格納されており、メモリ出力を切り替えることにより、異なる回路を構成することができる。

【0024】マルチコンテキスト技術は、回路の再構成時間を大幅に短縮することができる。また、回路を再構成するための回路情報をプログラマブル論理回路に転送するときの入出力バスラインの負荷を小さくできるため、高速化と低消費電力化に有利な構成である。しかし、回路情報を格納するためのメモリ回路領域をプログラマブル論理回路と一体構成する必要があり、集積回路化した場合のプログラマブル論理回路全体の回路規模が大きくなるという欠点を有している。

【0025】

【発明が解決しようとする課題】本発明は、上述した事情に鑑みてなされたもので、プログラマブル論理回路を再構成するためのメインプロセッサの負荷を大幅に軽減させることができ、これにより低消費電力化を図るとともに、メインプロセッサを有効に活用できるようにしてシステム全体のパフォーマンスを向上させた情報処理システムを提供することを目的とするものである。

【0026】

【課題を解決するための手段】一般にアプリケーションプログラムにおいて、処理のフローが決定されていて、そのために用いる処理回路が複数の種類の組み合わせであらかじめ選択指定可能な場合、同じ順番を単位として一定の繰り返される処理の場合などには、あらかじめ使用する回路情報と順番を決定できる。本発明はこれを利用し、プログラマブル論理回路に処理回路を構成するための回路情報を、予め順番に回路情報保持手段に保持させておく。そしてCPUなどの制御手段は、プログラマブル論理回路で処理を行う処理データのヘッダ部などに、使用する回路情報を特定するための情報を含めて、処理を依頼する。インタフェース手段は、与えられた処理データの例えばヘッダ部などから、プログラマブル論理回路の再構成に使用する最初の回路情報を特定するとともに、特定された回路情報から回路情報保持手段に保持されている順番に回路情報を取り出し、プログラマブル論理回路を再構成して処理回路を構成し、構成した処理回路に処理データの処理を行わせる。

【0027】このようにして、プログラマブル論理回路では、予め設定されている順番で、処理回路の再構成と実行が自動的に行われる。そのため、CPUなどの制御

手段では、最初に行う指示を行うだけで、以後順番に行われるプログラマブル論理回路の再構成や実行指示を行う必要がない。これによってCPUなどの制御手段によるプログラマブル論理回路の制御を行うための処理ステップが大幅に削減され、高速化を図ることができるとともに、消費電力を大幅に低減することができる。これとともに、プログラマブル論理回路を含めた情報処理システムにおける全体のパフォーマンスを向上させることができる。

【0028】

【発明の実施の形態】図1は、本発明の情報処理システムの実施の一形態を示す構成図である。図中、1は情報処理システム、2はネットワーク、3は外部機器、11はCPU、12はホストバス、13はチップセット、14はメインメモリ、15はシステムバス、16はハードディスクインタフェース、17はハードディスクドライブ、18は通信インタフェース、19は拡張ハードウェア部、21はプログラマブル論理回路、22はローカルメモリ、23はプログラマブル論理回路インタフェース、24は外部入出力インタフェースである。

【0029】情報処理システム1において、CPU11のホストバス12に、チップセット13に含まれる図示しないメモリコントローラを介して、DRAMで構成されるメインメモリ14が接続されている。ホストバス12は、チップセット13に含まれる図示しないバスブリッジを介してシステムバス15に接続されている。システムバス15は、例えばPCIバスやISAバス、その他各種のバスを使用することができる。例えばシステムバス15としてPCIバスを用いた場合、チップセット13内にはホスト-PCIブリッジを備えていればよい。

【0030】システムバス15には、この例では、ハードディスクインタフェース16を介してハードディスクドライブ17が接続され、また通信インタフェース18を介してネットワーク2と接続されている。さらに拡張ハードウェア部19が接続され、この拡張ハードウェア部19を介して外部機器3が接続されている。もちろん他の種々の機器が直接あるいはインタフェースを介してシステムバス15に接続されていてもよく、また、ハードディスクおよびネットワークについても接続は任意である。

【0031】この例では、ハードディスクドライブ17にはアプリケーションプログラムが格納されている。アプリケーションプログラムは、ハードディスクインタフェース16、システムバス15、および、チップセット13に含まれる図示しないバスブリッジを介して、ハードディスクドライブ17からメインメモリ14にロードされてCPU11によって実行される。

【0032】また、通信インタフェース18は、LANやインターネットなどのネットワーク2を介して、様々

な機器との間でデータの転送を行うことができる。CPU 11によって実行されるアプリケーションプログラムは、この通信インタフェース 18を介して通信を行うことにより、ネットワーク 2に接続される例えば記憶装置に格納されている情報へのアクセスを行うことができ、様々なアプリケーションプログラムやデータなどを入手できる。この場合、ネットワーク 2に接続される通信インタフェース 18を介して転送したアプリケーションプログラムをメインメモリ 14に格納して実行したり、あるいはシステムバス 15から直接プログラマブル論理回路インタフェース 23を介してプログラマブル論理回路 21へ転送することもできる。

【0033】拡張ハードウェア部 19は、プログラマブル論理回路 21、ローカルメモリ 22、プログラマブル論理回路インタフェース 23を有している。またこの例では、外部入出力インタフェース 24もこの拡張ハードウェア部 19に設けられている。プログラマブル論理回路 21は、回路情報を変更することによって機能を随時変更し再構成することが可能であり、プログラマブル論理回路インタフェース 23を介してシステムバス 15に接続されている。また、この例では外部入出力インタフェース 24を介して外部機器 3とも接続されており、プログラマブル論理回路 21における処理により、外部機器 3を制御可能に構成した例を示している。

【0034】ローカルメモリ 22は、プログラマブル論理回路 21で処理を行う処理データや、処理後の中間処理データなどを保持することができる。また、予め使用する回路情報と順番を保持する回路情報保持手段としても機能する。

【0035】プログラマブル論理回路インタフェース 23は、システムバス 15によってCPU 11やメインメモリ 14、ローカルメモリ 22、プログラマブル論理回路 21との間でデータ転送や制御を行うためのものである。またプログラマブル論理回路インタフェース 23は、転送されてきた処理データからプログラマブル論理回路の再構成に使用する最初の回路情報を特定する機能を有している。この最初の回路情報が例えば処理データのヘッダ情報として付加されている場合、処理データのヘッダ情報を解釈することによって実現できる。また、回路再構成のための回路情報や処理データ、中間処理データの最後に付加されるEOFのマーカ検出機能なども含まれている。さらに、特定した最初の回路情報をもとにローカルメモリ 22から回路情報を順番に取り出し、順次プログラマブル論理回路 21を再構成して処理回路を構成し、構成した処理回路に処理データの処理を行わせる。また、ローカルメモリ 22とプログラマブル論理回路 21との間での処理データや中間処理データの転送も行う。

【0036】なお、予め使用する回路情報等は、例えばメインメモリ 14やハードディスクドライブ 17に格納

しておいてもよい。この場合、プログラマブル論理回路インタフェース 23は、メインメモリ 14やハードディスクドライブ 17に対して、CPU 11を動作させずにデータ転送を行う機能を有していればよい。例えばメインメモリ 14とローカルメモリ 22を同じメモリ空間においてアクセス可能に構成し、いずれのメモリを利用しているかを意識しないでアクセスできるように構成してもよい。

【0037】また、拡張ハードウェア部 19は、集積化により一体化することで、入出力部のライン負荷を低減させたり、専用バス化の構成により、高速化及び低消費電力化を図ることができる。

【0038】図2は、プログラマブル論理回路の一例を示す平面構造図、図3は、同じく内部構造の一例を示すブロック図である。図中、31は論理セル、32は配線領域、33は入出力端子、41はコンフィギュレーションメモリ、42は回路素子である。プログラマブル論理回路 21は、回路情報を格納するためのコンフィギュレーションメモリ 41と、論理セル 31や配線領域 32からなる回路素子 42と、入出力端子 33とで構成されている。

【0039】コンフィギュレーションメモリ 41は、EEPROM、SRAMなどの書き換え可能なメモリ素子で構成されている。回路情報はアドレスとデータの対で構成される。コンフィギュレーションメモリ 41にアドレスを与えて、そのアドレスに対応するメモリセルにアドレスと対になったデータを格納すると、このデータに従って、論理セル 31内の回路構成や、論理セル 31と入出力端子 33を相互に接続する配線領域 32の接続状態が再構成される。コンフィギュレーションメモリ 41の一部分を書き換えることにより、プログラマブル論理回路 21が動作中であっても、回路を部分的に再構成することができる。

【0040】プログラマブル論理回路 21に再構成された回路素子 42に、入出力端子 33を介して処理すべきデータ（処理データや中間処理データ）が入力され、また、その処理結果（中間処理データ）が出力される。データ入力先の論理セルと、データ出力元の論理セルは、論理セルの位置に対応するセル座標を示した制御コードによってアプリケーションプログラムが指定する。

【0041】以上のシステム構成によって、プログラマブル論理回路 21による処理データの入出力方法から、以下のような処理形態があげられる。データとしては、マルチメディアのような画像、音声などのストリーミングデータなどがある。

① ネットワーク 2を経由して情報システム 1に入力したデータを、システムバス 15を経由して、プログラマブル論理回路 21に再構成した処理回路で処理する。

② ハードディスクドライブ 17などの記憶装置に格納されたデータを、システムバス 15を経由して、プロ

ラマブル論理回路 21 に再構成した処理回路で処理する。

③ 外部の記憶装置からの転送やアプリケーションの処理結果などのデータで、メインメモリ 14 に一時的に格納されたデータを、システムバス 15 を経由して、プログラマブル論理回路 21 に再構成した処理回路で処理する。

④ ローカルメモリに格納されたデータを、プログラマブル論理回路インタフェース 23 を経由して、プログラマブル論理回路 21 に再構成した処理回路で処理する。

【0042】次に、本発明の情報処理システムの実施の一形態における動作について説明する。アプリケーションプログラムにおいて、処理のフローが決定されていて、そのために用いる機能回路が複数の種類の組み合わせであらかじめ選択指定可能な場合や、同じ順番を単位として繰り返される処理の場合などには、あらかじめ使用する回路情報と順番を決定できる。そこで、アプリケーションの処理が開始される前に、予め、使用する回路情報と順番等の情報を参照テーブルとして例えばローカルメモリ 22 などに登録しておく。なお、全く不定な順番で、任意の処理の次にどのような処理を行うことになるのか特定できない場合には、メインプロセッサによって、その都度、制御を行えばよいので、ここでは対象外とする。

【0043】図 4 は、参照テーブルの一例の説明図である。図 4 に示した例では、処理回路の順番を示す処理ステップ No. i に対応して、再構成のための回路情報が格納されているメモリ中の開始アドレスを示すポインタ、中間処理を行うことになる場合に対応してアロケーションされた中間処理データを格納するメモリ中の開始アドレスを示すポインタ、プログラマブル論理回路 21 に同時に再構成可能な回路数の情報によって参照テーブルが構成されている。

【0044】アプリケーションの処理で i 番目の処理回路で処理した結果を次の i+1 番目で処理する場合には、i 番目の処理回路から出力されるデータは中間処理データとなる。また最後の M 番目で処理した結果は、最終出力データとなる。この中間処理データを格納するメモリの開始アドレスポインタが図 4 におけるポインタ P d i である。なお、最終データもさらに別のアプリケーションで用いられることも考えられるので、最終データも中間処理データ用メモリへ格納しておいてもよい。

【0045】また、再構成する処理回路について、プログラマブル論理回路の回路規模に応じて、同時に再構成できる回路数 N i は、i 番目の回路からの同時再構成可能最大数である。この N i に対応する回路の再構成のモードとして、1 個の場合には自動的に単独コンフィギュレーションモードになり、複数個の場合には単独コンフィギュレーションモードまたは連続コンフィギュレーションモードの選択ができる。これは、アプリケーション

開始時に CPU によって選択指定できる。

【0046】この図 4 に示すような参照テーブルは、アプリケーションが実行されるときに情報処理システムで用いられるプログラマブル論理回路 21 の品種や回路規模サイズ、メインメモリ 14 やローカルメモリ 22 などのメモリサイズなどを考慮して、CPU 11 によって作成する。あるいは、当然ながら、決まった処理を同じシステムで行う場合には、処理の実行の都度にこの参照テーブルを作成する必要はなく、予め用意しておくこともできる。

【0047】図 5 は、処理データおよび回路情報と、これらを格納するメモリアドレス空間の説明図である。図 4 に示すような参照テーブルの情報は、図 5 に示すように、例えば処理データのヘッダ部に付加しておくことができる。プログラマブル論理回路インタフェース 23 において、処理データを受け取った際にヘッダ部を解析し、これらの情報を取り出して、参照テーブルを作成することができる。また、このような参照テーブルをもとに、ある処理ステップ i を実行する際には、処理ステップ No. i に対応付けられている回路情報用のポインタ P c i と中間処理データ用のポインタ P d i が割り付けられる。処理の順番によっては、同じポインタを用いて上書きでアロケートすることによって、メモリの使用効率を向上させることもできる。

【0048】なお、図 5 に示すメモリアドレス空間は、ローカルメモリ 22 のメモリアドレス空間、あるいは、メインメモリ 14 とローカルメモリ 22 を一体としたメモリアドレス空間であってよい。後者の場合、例えば回路情報や中間処理データを格納する領域などがメインメモリ 14 上に確保される場合もある。

【0049】前述の回路情報と順番において、図 4 に示すように、処理ステップ No. i として最初の 0 番から M 番までの M+1 ステップの場合で、i 番目の処理を行う際の動作について述べる。再構成する回路情報を格納するメモリの開始アドレスポインタを P c i とする。また、中間処理データを格納するメモリの開始アドレスポインタを P d i とする。さらに、再構成する処理回路について、プログラマブル論理回路の回路規模に応じて、同時に再構成できる回路数を N i とする。

【0050】アプリケーションの処理データファイル及び回路情報を読み出し、あらかじめアプリケーションプログラムの実行前に、回路情報をメモリ空間に割り付けておく。また、CPU 11 によって前述の 4 種の情報をアプリケーションの処理データファイルのヘッダ部に付加する。なお、処理データファイルや回路情報は、例えば、ネットワーク 2 に接続された記憶装置、情報処理システム 1 内部のハードディスクドライブ 17 などの記憶装置、メインメモリ 14、あるいは、ローカルメモリ 22 など、いずれの記憶装置に記憶されていてもよい。

【0051】アプリケーションプログラムの実行が CPU

U11によって開始されると、処理データファイルがプログラマブル論理回路インタフェース23に転送されてヘッダ部に付加されている情報が解釈される。最初の回路情報によってプログラマブル論理回路21に対する処理回路の再構成が行われ、回路情報の最後を示すEOFのマーカにより、再構成が終了する。このEOFマーカをプログラマブル論理回路インタフェース23で検出し、ヘッダ部に続くデータ部の転送が行われて、再構成した処理回路におけるデータ処理が進められていく。

【0052】処理されたデータを次に再構成する処理回路で用いる場合には、処理されたデータを中間処理データとして、さきにメモリ空間にアロケートした領域に一旦格納する。そして、次の処理回路を再構成した後、中間処理データを新たに再構成した処理回路に入力し、データ処理を進める。最終データは、例えば外部入出力インタフェース24を介して外部機器3に転送されたり、あるいはメインメモリ14、ハードディスクドライブ17、ネットワーク2に接続された記憶装置などに転送される。または、中間処理データと同様に格納されているもよい。

【0053】以上のように、CPU11が最初の開始制御を行うことによって、回路の再構成とデータ処理が順次実行されていくので、CPU11はプログラマブル論理回路21の再構成のための処理を行う必要がない。そのため、CPU11の負荷を軽減して消費電力を大幅に低減し、またCPU11とプログラマブル論理回路21との並行動作を可能として情報処理システムにおける全体のパフォーマンスを向上させることができる。

【0054】次に、プログラマブル論理回路21への処理回路の再構成領域とそれぞれの処理回路によって処理された中間処理データのメモリへの格納あるいは連続処理について、以下、3つの形態をあげて説明する。

【0055】図6は、プログラマブル論理回路への処理回路の第1の再構成例の説明図、図7は、同じく動作時の一例を示すタイミングチャートである。図6に示す例では、シングルタスクモードで、処理回路の再構成とその処理回路での処理の実行を順次行いながら、中間処理データの入出力をその間に入れていく方式を示している。この例は、プログラマブル論理回路21の構成可能な最大回路規模と再構成する回路規模との関係において、連続する処理回路が同時に再構成できる領域が確保できない場合に対応する形態である。

【0056】最初の処理回路の再構成が完了し、データ処理が行われて得られた結果は、一時的にプログラマブル論理回路21の内部メモリまたはローカルメモリ22やメインメモリ14など外部メモリを利用して中間処理データのまゝ格納する。そして、最初に再構成された領域に上書きして次の処理回路の再構成を行う。次の処理回路の再構成が完了すると、続けて中間処理データに対してデータ処理を行う。このようにしてデータ処理が継

続されていく。処理モードとしては、回路の同時再構成可能な数Niは、1個である単独コンフィギュレーションモードの場合である。

【0057】一例として、図6に示す例について、図7に示すタイミングチャートを用いて説明する。まず、CPU11によって、例えば図4に示すような参照テーブルが用意され、図5に示すようにメモリアドレス空間が割り当てられる。プログラマブル論理回路インタフェース23は、CPU11から処理データが転送されてくると、その処理データのヘッダ部に付加された情報を解釈して、プログラマブル論理回路21に対する回路Aの再構成を開始する(図7-①)。

【0058】回路Aの再構成が完了すると、処理データを回路Aに入力し、回路Aを用いた処理が開始される(図7-②)。回路Aによって得られた結果は、中間処理データとしてメモリに格納されていく(図7-③)。

【0059】処理データのEOFが検出されて処理が終了すると、次の回路Bの再構成を行う(図7-④)。最初に再構成された回路Aの領域に上書きして次の処理回路である回路Bの再構成を行うことができる。図6は、この回路Bを再構成したときの状態を示している。回路情報のEOFが検出されると、回路Bの再構成が完了する(図7-⑤)。回路Bの再構成が完了すると、回路Bは中間処理データに対してデータ処理を開始する(図7-⑥)。

【0060】このステップが繰り返されて、処理ステップMで終了する。このように、最初にCPU11による制御を施すだけで、あとは、処理回路の再構成と、再構成された処理回路における処理の実行が、あらかじめ設定された規定通りに連続して行われる。これによって、CPU11の制御負荷が大幅に軽減され、消費電力が大幅に低減できる。

【0061】図8は、プログラマブル論理回路への処理回路の第1の再構成例における動作の一例を示すフローチャートである。上述の例について処理動作をまとめると図8に示すようになる。S101において処理ステップを示す変数iを0に初期化し、S102において、処理データを入力し、ヘッダ部の情報を解釈してポインタ等の設定を行う。

【0062】S103において、回路情報Pciにアクセスしてプログラマブル論理回路21に対して処理回路の再構成を行う。S104において回路情報の終了を示すEOFを検出すると、処理回路の再構成を終了する。そしてS105において、中間処理データPd(i-1)へアクセスして再構成した処理回路に入力し、あるいは、処理データを再構成した処理回路に入力して、S106においてデータ処理を開始する。

【0063】S107において、処理データあるいは中間処理データの終了を示すEOFを検出すると、処理回路における処理を終了する。S108において、処理ス

テップを示す変数 i に 1 を加え、S109において、変数 i の値が M 以下か否かを判定する。変数 i の値が M 以下であれば、S110において、処理結果を中間処理データとして中間処理データ用のメモリ領域 $Pd(i-1)$ に格納する。そして S103 へ戻り、次の処理回路の再構成および処理を繰り返す。

【0064】処理ステップ M までの処理を行い、処理ステップを示す変数 i の値が M を超える場合には、S111において、処理後のデータを最終データとして出力し、プログラマブル論理回路 21 における処理を終える。

【0065】なお、上述の説明では中間処理データを生成して処理する形態で記述したが、図 2 に示すように回路 A の結果を内部メモリで受けて、引き続き回路 B で処理する場合には、内部メモリの代わりに、データラッチ回路を用いた一時的なデータ保持方法の利用も可能である。

【0066】図 9 は、プログラマブル論理回路への処理回路の第 2 の再構成例の説明図、図 10 は、同じく動作時の一例を示すタイミングチャートである。図 9 に示す例では、シングルタスクモードではあるが、複数の処理が同時に形成できる場合に、複数の処理回路の再構成を行ってから、処理を連続して実行する形態である。処理モードとして、回路の同時再構成可能な数 N_i までの再構成を連続的に行う、連続コンフィギュレーションモードの選択である。

【0067】プログラマブル論理回路 21 の構成可能な最大回路規模と再構成する処理回路の回路規模との関係において、連続する処理回路が同時に再構成できる領域が確保できるが、データ処理が行われて得られた結果は、一時的にプログラマブル論理回路の内部メモリまたはローカルメモリやメインメモリの外部メモリを利用して中間処理データのまま格納して行っていく場合である。ただし、中間処理データを介しながら分割して処理を進めていく場合も含めることができる。当然ながら、処理モードとして、処理回路を一つずつ再構成してゆく単独コンフィギュレーションモードも選択できる。この場合には、図 6 に示した形態として扱うこともできる。

【0068】一例として、図 9 に示す例について、図 10 に示すタイミングチャートを用いて説明する。まず、CPU11 によって、図 4 に示すような参照テーブルが用意され、図 5 に示すようにメモリアドレス空間が割り当てられる。プログラマブル論理回路インタフェース 23 は、CPU11 から処理データが転送されてくると、その処理データのヘッダ部に付加された情報を解釈する。そして、プログラマブル論理回路 21 に再構成可能な複数の回路、ここでは回路 A と回路 B の再構成を開始する (図 10-①、②)。

【0069】2 つの回路情報の EOF が検出されると再構成を完了し、これらの処理回路を用いた処理の実行が

開始される (図 10-③、④)。ここでは、回路 A によって処理された結果が回路 B に入力され、回路 B による処理結果が中間処理データとしてメモリに格納されている。ここでは、最後の処理回路による処理結果のみを示している (図 10-⑤、⑥)。

【0070】このような処理が繰り返されて、処理ステップ M で終了する。このように連続処理が可能になるため、中間処理データのすべてに対してメモリへ格納せずに処理を進めることができる。そのため、中間処理データを格納する回数が低減され、高速処理、消費電力の低減が図れる。また、最初に CPU11 による制御を施すだけで、あとは、処理回路の再構成と、再構成された処理回路における処理の実行が、あらかじめ設定された規定通りに連続して行われる。これによって、CPU11 の制御負荷が大幅に軽減され、消費電力が大幅に低減できる。

【0071】図 11 は、プログラマブル論理回路への処理回路の第 2 の再構成例における動作の一例を示すフローチャートである。上述の第 2 の構成例について処理動作をまとめると図 11 に示すようになる。S121 において処理ステップを示す変数 i を 0 に初期化し、S122 において、処理データを入力し、ヘッダ部の情報を解釈してポインタ等の設定を行う。

【0072】S123 において、再構成した回路数を示す変数 j を 0 に初期化する。S124 において、回路情報 $Pc(i+j)$ にアクセスしてプログラマブル論理回路 21 に対して処理回路の再構成を行う。S125 において回路情報の終了を示す EOF を検出すると、処理回路の再構成を終了する。S126 において変数 j の値に 1 を加算し、S127 において、新たな変数 j の値が処理ステップ i における同時構成可能な回路数 N_i より小さいか否かを判定する。新たな変数 j の値が N_i より小さければ、S124 へ戻って、同時に再構成可能な次の処理回路の再構成を行う。変数 j の値が N_i に達すると、S128 において、同時に再構成可能な回路数までの処理回路の再構成を終了する。

【0073】S129 において、変数 i が 0 の場合には処理データを、また変数 i が 0 でない場合には中間処理データ $Pd(i+N_i-1)$ へアクセスして、再構成した処理回路に入力し、S130 においてデータ処理を開始する。

【0074】S131 において、処理データあるいは中間処理データの終了を示す EOF を検出すると、処理回路における処理を終了する。S132 において、処理ステップを示す変数 i に N_i を加え、S133 において、新たな変数 i の値が M 以下か否かを判定する。変数 i の値が M 以下であれば、S134 において、処理結果を中間処理データとして中間処理データ用のメモリ領域 $Pd(i+N_i-1)$ に格納する。そして S123 へ戻り、次の複数の処理回路の再構成および再構成した複数の処

理回路による連続処理を繰り返す。

【0075】処理ステップMまでの処理を行い、処理ステップを示す変数*i*の値がMを超える場合には、S135において、処理後のデータを最終データとして出力し、プログラマブル論理回路21における処理を終える。

【0076】なお、上述の第1の再構成例と同様に、シングルタスクモードで単独コンフィギュレーションを行う場合には、それぞれの処理ステップごとに中間処理データをメモリへ格納していけばよい。連続した回路の同時構成が可能のため、連続した処理が可能になる。当然、必要に応じて中間処理データはローカルメモリ22などへ格納されてもよい。

【0077】図12は、プログラマブル論理回路への処理回路の第3の再構成例の説明図、図13は、同じく動作時の一例を示すタイミングチャートである。図12に示す例では、マルチタスクモードとして、処理回路の再構成が完了すると処理が実行されるとともに、その処理の実行中に並行して次の処理回路の再構成がプログラマブル論理回路の別の領域に対して行われる。そして、先の処理回路による処理結果が連続して処理されたり、別のデータによる並列処理などが行われていく。例えば回路Aによるプロセス中に回路Bのコンフィギュレーションを行って処理を連続的に行っていくものである。しかも、処理回路の再構成は、プログラマブル論理回路21に同時再構成可能な数*N_i*までの処理回路の再構成を連続的に行う、連続コンフィギュレーションモードが選択される。

【0078】なおこの例では、さきの処理回路の再構成が完了してデータ処理が行われているときに、次の処理回路の再構成が並列して行われていく。さきの処理回路によるデータ処理の結果は、一時的にプログラマブル論理回路21の内部メモリまたはローカルメモリ22やメインメモリ14等の外部メモリを利用して中間処理データのまま格納して、次の回路の再構成が完了したときに続けてデータ処理が継続されていく。しかし、次の処理回路の再構成が、さきの処理回路の出力までに完了する場合には、図9に示したように回路Aの出力を直接、回路Bに入力するように構成してもよい。

【0079】一例として、図12に示す例について、図13に示すタイミングチャートを用いて説明する。まず、CPU11によって、図4に示すような参照テーブルが用意され、図5に示すようにメモリアドレス空間が割り当てられる。プログラマブル論理回路インタフェース23は、CPU11から処理データが転送されてくると、その処理データのヘッダ部に付加された情報を解釈する。プログラマブル論理回路インタフェース23は、CPU11から処理データが転送されてくると、その処理データのヘッダ部に付加された情報を解釈して、プログラマブル論理回路21に対する回路Aの再構成を開始

する(図13-①)。

【0080】回路Aに関する回路情報のEOFを検出し、回路Aの再構成が完了すると、処理データを回路Aに入力し、回路Aを用いた処理が開始される(図13-②)。回路Aによって得られた結果は、中間処理データとしてメモリに格納されていく(図13-③)。

【0081】この回路Aにおける処理と並行して、次の回路Bの再構成を開始する(図13-④)。このような処理回路において処理を行っている間に、プログラマブル論理回路に回路の同時再構成可能な数*N_i*までの再構成を連続的に行うことができる。

【0082】回路Aにおける処理において処理データのEOFを検出して処理の終了を検出するとともに、次の回路Bの再構成の終了を検出すると、回路Bにおける中間処理データを用いた処理の実行が開始される(図13-⑤、⑥)。

【0083】このように、処理回路の再構成と、再構成した処理回路の実行の並列処理によって、処理回路の再構成による処理の待ち時間を短くして高速化を図ることができるとともに、処理を連続して実行することによる処理の高速化を図ることができる。また、最初にCPU11による制御を施すだけで、あとは、処理回路の再構成とその実行を、あらかじめ設定された規定通りに連続して行うことができる。これによって、CPU11の制御負荷が大幅に軽減され、消費電力が大幅に低減できる。

【0084】図14は、プログラマブル論理回路への処理回路の第3の再構成例における動作の一例を示すフローチャートである。上述の第3の構成例について処理動作をまとめると図14に示すようになる。S141において処理ステップを示す変数*i*を0に初期化し、S142において、処理データを入力し、ヘッダ部の情報を解釈してポインタ等の設定を行う。

【0085】S143において、回路情報*P_{ci}*にアクセスしてプログラマブル論理回路21に対して処理回路の再構成を行う。S144において回路情報の終了を示すEOFを検出すると、処理回路の再構成を終了する。そしてS145において処理ステップ*i*の処理回路の再構成が終了していることを確認後、S146において、中間処理データ*P_{d(i-1)}*へアクセスして再構成した処理回路に入力し、あるいは、処理データを再構成した処理回路に入力して、S147においてデータ処理を開始する。

【0086】このような再構成された処理回路における処理の実行と並行して、S161以降の処理回路の再構成を行う。まずS161において変数*i*の値を変数*k*に待避し、S162において、再構成した回路の処理ステップを示す変数*j*の値を*k+1*に初期化する。そして、S163において、回路情報*P_{c(j)}*にアクセスしてプログラマブル論理回路21に対して、空いている領域

に処理回路の再構成を行う。S164において回路情報の終了を示すEOFを検出すると、処理回路の再構成を終了する。S165において変数jの値に1を加算し、S166において、同時に再構成した回路数が、処理ステップkにおいて同時構成可能な回路数 N_k より小さいか否かを判定する。この判定のために、いままで再構成した回路数+1を示す新たな変数jの値と、 $k+N_k$ の値とを比較する。新たな変数jの値が $k+N_k$ より小さければ、S163へ戻って、同時に再構成可能な次の処理回路の再構成を行う。変数jの値が $k+N_k$ に達すると、同時に構成可能な回路数に達したものと、処理回路の再構成を一旦停止する。

【0087】S167において、変数jの値がMに達していれば、すべての処理回路の再構成が終了したものと、再構成の処理を終了する。変数jの値がM以下であれば処理を継続し、S168において、再構成が終了している処理回路のすべてにおいて処理が終了するまで待ち、その後、S161へ戻って、次の同時に構成可能な回路数だけの処理回路の再構成処理を行う。

【0088】このような処理回路の再構成処理と並行して行われていた処理回路における処理の実行は、S148において処理データあるいは中間処理データの終了を示すEOFを検出して終了する。S149において、処理ステップを示す変数iに1を加え、S150において、新たな変数iの値がM以下か否かを判定する。変数iの値がM以下であれば、S151において、処理結果を中間処理データとして中間処理データ用のメモリ領域Pd(i-1)に格納する。そしてS145へ戻り、次の処理回路の再構成終了を確認してから、次の処理回路による連続処理を繰り返す。

【0089】処理ステップMまでの処理を行い、処理ステップを示す変数iの値がMを超える場合には、S152において、処理後のデータを最終データとして出力し、プログラマブル論理回路21における処理を終える。

【0090】なお、上述の説明では、処理回路の再構成を、同時に構成可能な回路数ごとに行っている。しかしこれに限らず、例えば処理が終了した処理回路の領域を順次開放してゆき、次の処理回路が構成可能な領域が確保できた時点ですぐに次の処理回路を再構成するように構成してもよい。

【0091】上述の3つの例は、適宜組み合わせることが可能である。図15は、プログラマブル論理回路への処理回路の第4の再構成例の説明図、図16は、同じく動作時の一例を示すタイミングチャートである。この例では、最初に上述の第2の再構成例で示したように、同時に構成可能な回路数だけの処理回路の再構成を先に行っておき、その後、第3の再構成例で示したように、処理が終了した処理回路の領域を開放して新たな処理回路の再構成を行うことができる。

【0092】一例として、図15に示す例について、図16に示すタイミングチャートを用いて説明する。まず、図15(A)に示すように、プログラマブル論理回路21に再構成可能な複数個の回路、ここでは回路Aと回路Bの再構成を開始する(図16-①、②)。

【0093】2つの回路情報のEOFが検出されると再構成を完了し、これらの処理回路を用いた処理の実行を開始する(図16-③、④)。ここでは、回路Aによって処理された結果が回路Bに入力され、回路Bによる処理結果が中間処理データとしてメモリに格納されていく(図16-⑤、⑥)。

【0094】回路Aの処理が終了した時点で、回路Aが配置されていたプログラマブル論理回路21内の領域は不要となる。そのため、回路Bの処理実行中に並行して、回路Aが構成されていた領域も用いて、回路Cの再構成を開始する(図16-⑦)。このような処理回路の再構成は、処理回路を配置可能であればいくつでも行ってよい。このようにして、図15(B)に示すように、回路Aが配置されていた領域も用いて回路Cの再構成が行われる。

【0095】回路Bにおける処理において処理データのEOFを検出して処理の終了を検出するとともに、次の回路Cの再構成の終了を検出すると、回路Bから出力された中間処理データを用いて、回路Cにおける処理の実行が開始される(図16-⑧、⑨)。このような処理が繰り返されて、処理ステップMまでの処理を連続して行うことができる。

【0096】なお、上述の説明では、図1に示した構成をもとにして説明してきた。本発明の情報処理システムは図1に示す構成に限られることはなく、種々の変形が可能である。そのうちのいくつかの変形例については既に述べた。さらに極端な例として、例えば拡張ハードウェア部19の部分のみの構成や、さらには外部入出力インタフェース24も設けず、プログラマブル論理回路インタフェース23を介してのみデータの入出力を行う構成などであってもよい。

【0097】

【発明の効果】以上の説明から明らかなように、本発明によれば、あらかじめ使用する回路と順番を規定できる場合には、メインプロセッサによる最初のデータ処理開始の制御だけで、あとは処理データ(例えば処理データのヘッダ情報)を解釈するだけで、プログラマブル論理回路の複数の機能回路によるデータ処理を順次実行してゆくことができる。このため、メインプロセッサによる制御の処理ステップを大幅に削減して高速化を図ることができるとともに、消費電力も大幅に低減することができる。

【0098】また、回路の再構成のための制御に対するメインプロセッサの負荷が大幅に軽減されることによって、その処理パワーを他の処理へ使用できるため、シス

テム全体のパフォーマンスの向上を図ることができるという効果がある。

【図面の簡単な説明】

【図 1】 本発明の情報処理システムの実施の一形態を示す構成図である。

【図 2】 プログラマブル論理回路の一例を示す平面構造図である。

【図 3】 プログラマブル論理回路の内部構造の一例を示すブロック図である。

【図 4】 参照テーブルの一例の説明図である。

【図 5】 処理データおよび回路情報と、これらを格納するメモリアドレス空間の説明図である。

【図 6】 プログラマブル論理回路への処理回路の第 1 の再構成例の説明図である。

【図 7】 プログラマブル論理回路への処理回路の第 1 の再構成例における動作時の一例を示すタイミングチャートである。

【図 8】 プログラマブル論理回路への処理回路の第 1 の再構成例における動作の一例を示すフローチャートである。

【図 9】 プログラマブル論理回路への処理回路の第 2 の再構成例の説明図である。

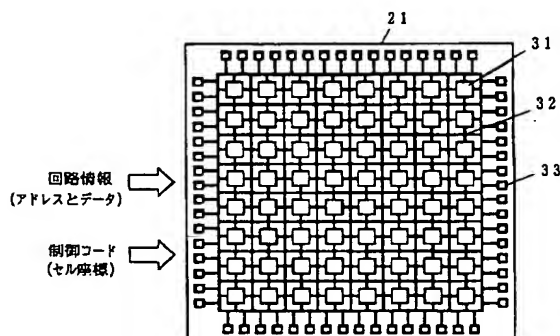
【図 10】 プログラマブル論理回路への処理回路の第 2 の再構成例における動作時の一例を示すタイミングチャートである。

【図 11】 プログラマブル論理回路への処理回路の第 2 の再構成例における動作の一例を示すフローチャートである。

【図 12】 プログラマブル論理回路への処理回路の第 3 の再構成例の説明図である。

【図 13】 プログラマブル論理回路への処理回路の第 3 の再構成例における動作時の一例を示すタイミングチャートである。

【図 2】



ャートである。

【図 14】 プログラマブル論理回路への処理回路の第 3 の再構成例における動作の一例を示すフローチャートである。

【図 15】 プログラマブル論理回路への処理回路の第 4 の再構成例の説明図である。

【図 16】 プログラマブル論理回路への処理回路の第 4 の再構成例における動作時の一例を示すタイミングチャートである。

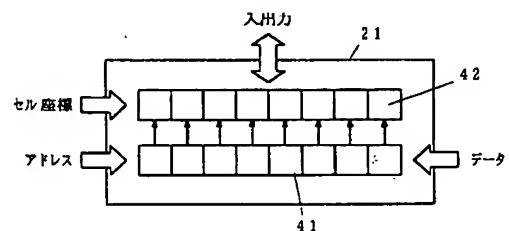
【図 17】 従来の情報処理システムの一例を示すブロック図である。

【図 18】 DPGA の論理セル構造の一例の説明図である。

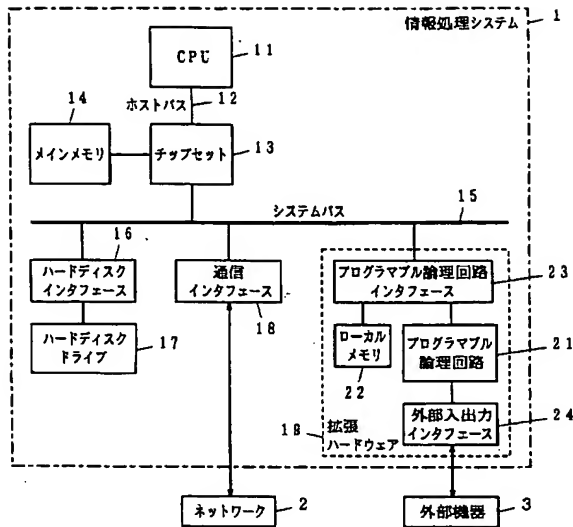
【符号の説明】

1…情報処理システム、2…ネットワーク、3…外部機器、11…CPU、12…ホストバス、13…チップセット、14…メインメモリ、15…システムバス、16…ハードディスクインタフェース、17…ハードディスクドライブ、18…通信インタフェース、19…拡張ハードウェア部、21…プログラマブル論理回路、22…ローカルメモリ、23…プログラマブル論理回路インタフェース、24…外部入出力インタフェース、31…論理セル、32…配線領域、33…入出力端子、41…コンフィギュレーションメモリ、42…回路素子、51…アプリケーションサーバ、52、53…クライアントコンピュータ、54…通信ネットワーク、55…メインプロセッサ、56…拡張ハードウェア、57…アプリケーションプログラム、58…拡張コード、59…メインプロセッサコード、60…OS、61…コード選択機能、71…DRAM、72、75…マルチプレクサ、73…ルックアップテーブル、74…フリップフロップ。

【図 3】



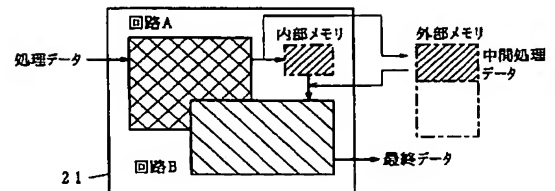
【図 1】



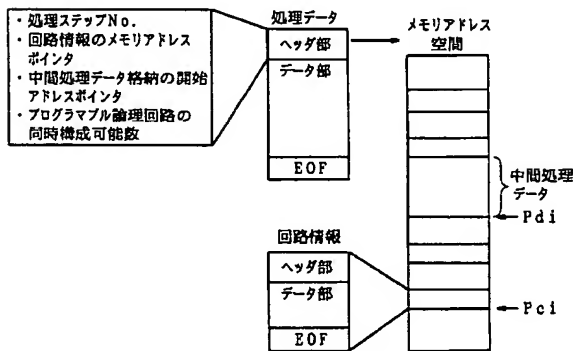
【図 4】

処理 ステップ No.	再構成する回路情報 を格納するメモリの 開始アドレスポイント	中間処理データを 格納するメモリの 開始アドレスポイント	プログラマブル論理回路 への同時構成可能 回路数
0	Pc0	Pd0	N0
1	Pc1	Pd1	N1
2	Pc2	Pd2	N2
...
i	Pci	Pdi	Ni
...
M	PcM	PdM	NM

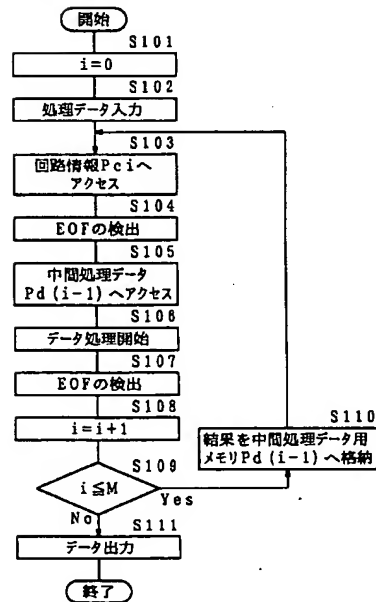
【図 6】



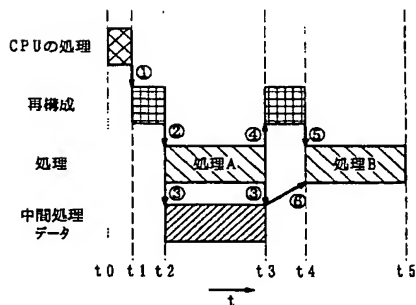
【図 5】



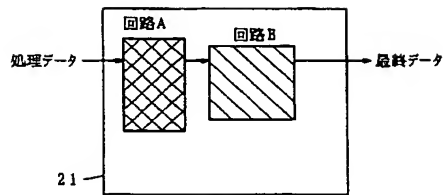
【図 8】



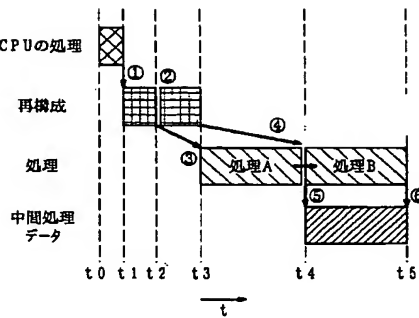
【図 7】



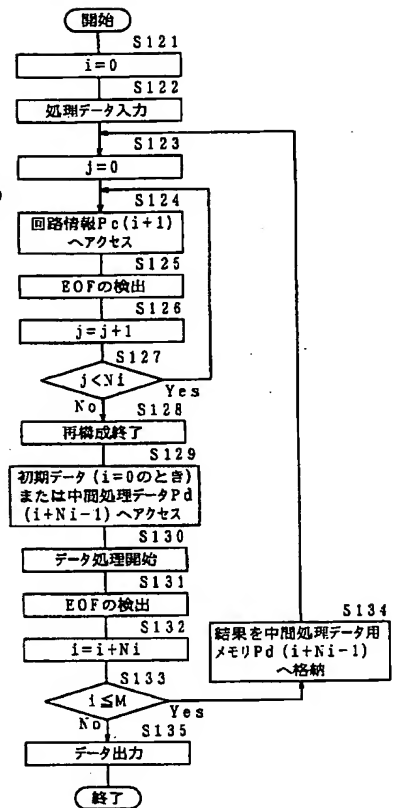
【図 9】



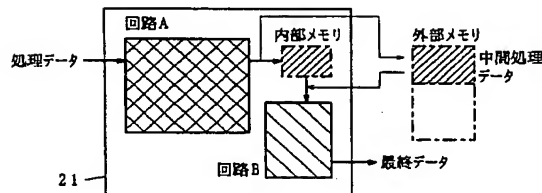
【図 10】



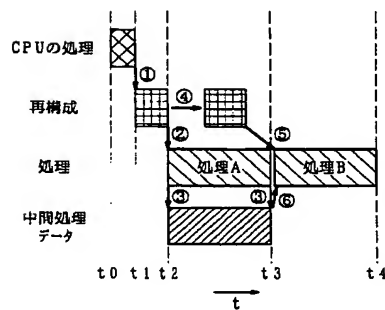
【図 11】



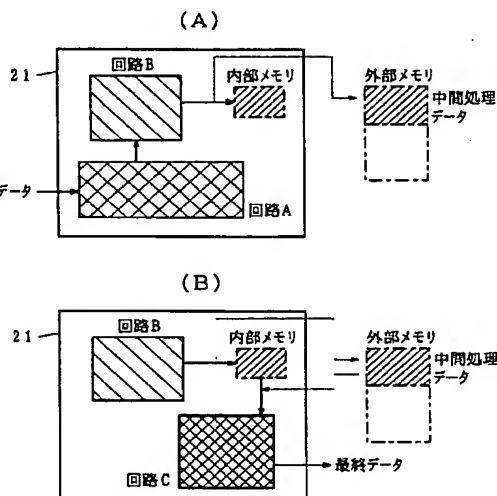
【図 12】



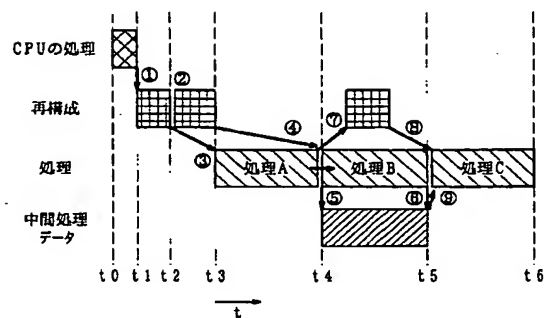
【図 13】



【図 15】



【図 16】



【图 18】

